software systems of Figure 2 lie below 100 tuples. These conclusions are in line with our measurements of the IFS/2 when supporting the CLIPS Production Rule System (see below).

(b) For the particular join tests of Figure 2, there is no advantage in having more than 27 search modules in the IFS/2 hardware for cardinalities under about 3000 tuples. Above that size, the IFS/2 performance curve can be kept linear by adding more search modules.

For production rule systems, we may represent facts (i.e. Working Memory Elements) and the left-hand side of rules (i.e. Condition Elements) as IFS tuples of constants and wild cards. This permits the use of the IFS/2's associative memory and its relational algebraic capability to speed up the time-consuming match phase of production systems. To test this scheme, we have re-programmed the run-time software of the CLIPS production system so as to interface it with the IFS/2 [22]. CLIPS (C Language Integrated Production System) is an OPS5-like system developed by the NASA Johnson Space Research Center. We have carried out measurements on two versions of two simple CLIPS synthetic test programs whose common static characteristics are: three Condition Elements per rule; a maximum of two shared and two unshared variables per rule; 100 rules. The right-hand ('action') side of each rule was arranged to cause a change to the working memory which subsequently matched with a left-hand-side CE containing either one or two shared variables. The two versions, V1, V2, of the two synthetic programs P1, P2, are distinguished thus:

<table>
<thead>
<tr>
<th>program</th>
<th>no. of attributes/ rule</th>
<th>WM-change matches a CE with:</th>
</tr>
</thead>
<tbody>
<tr>
<td>P1 V1</td>
<td>3</td>
<td>1 shared var.</td>
</tr>
<tr>
<td>P1 V2</td>
<td>3</td>
<td>2 shared vars.</td>
</tr>
<tr>
<td>P2 V1</td>
<td>6</td>
<td>1 shared var.</td>
</tr>
<tr>
<td>P2 V2</td>
<td>6</td>
<td>2 shared vars.</td>
</tr>
</tbody>
</table>

Table3

The tests were performed for given numbers of initial facts (Working Memory Elements) in the range 10 to 100,000 facts. The results, described more fully in [22], are plotted in Figure 3 on a log/log scale to show the speed-up factor of IFS-CLIPS over standard CLIPS. For this exercise, 'standard' CLIPS execution-times were measured on a Sun Sparc Workstation running at 24Mhz and having 16Mbytes of RAM.

It is seen from Figure 3 that use of the IFS/2 may be expected to speed CLIPS execution times by three orders of magnitude when there are several tens of thousand initial facts. Conversely, the IFS/2 actually slows down performance for production systems having fewer than about 100 initial facts. This is in line with the observations on Figure 2 given previously.

In addressing the problems of slow and complex software, we have introduced a systems architectural framework that allows direct hardware support for a useful range of primitive operations which occur frequently in non-numeric (e.g. symbolic) applications. The hardware takes the form of an add-on active memory unit called the IFS/2, in which SIMD techniques are employed to exploit parallelism in a manner that requires no effort on the part of the applications programmer. The actual implementation of the active memory unit is, however, not the primary concern of this paper. The important point is that the high-level (i.e. problem-solving) requirements of applications programmers have been recognised in terms of well-used bulk data types. A representational formalism has been devised, and whole-structure operations embedded in a procedural interface to a low-level supporting unit that can be implemented in a cost-effective manner. A software simulator of the IFS/2 active memory has been distributed to several institutions. We are currently evaluating a three-node, 27 search module, IFS/2 hardware prototype at Essex. Figures for relational join and for the match phase of a production rule system have been obtained for the IFS/2, when attached as a performance accelerator to a standard Sun Workstation. The results indicate that the IFS/2 has the potential to increase the speed of whole-structure operations by a useful amount.

10. Acknowledgements.

It is a pleasure to acknowledge the contribution of other members of the IFS team at Essex. Particularly, Jenny Embly and Andy Marsh contributed to the IFS/2 hardware design; Jiwei Wang was responsible for an earlier version of the software simulator; Edward Tsang has contributed to the definition of graph operations; Shutian Lin was responsible for measurements on the CLIPS production system. The work described in this paper has been supported by SERC grants GR/F/06319, GR/F/61028 and GR/G/30867.

11. References.


FIGURE 1: Worst-case member search: IFS/2 compared with a Distributed Array Processor.
FIGURE 2: Elapsed time for join, showing the performance of the IFS/2 hardware when compared with six software systems.
FIGURE 3: Speed-up of IFS-CLIPS versus standard CLIPS